The Open University

MT365
Audio Notes 1

CDA5674 pp 4 - 33

CDA 5675 pp 36 - end

Copyright © 1996 The Open University

SUP 32821 0
2.5

This booklet contains the printed material for use with Audio-tapes 1 and 2, which go with Blocks 1 and 2.

You will need to play the tape at the same time as you study the frames on the following pages.

Place the cassette player within easy reach. There are points on the tape where we have indicated that we want you to stop the tape and do some work for yourself, but you will probably find it necessary to stop the tape more often than this. Indeed, you should get into the habit of frequently stopping the tape and giving yourself time to think.

Make sure that you have paper and pencil handy before starting each tape sequence.

BLOCK 1

Notes for Networks 1

There are two sequences on the tape. The heading numbers 3.3 and 4.1 below refer to the corresponding sections in *Networks 1*.

In each sequence we describe an algorithm for solving a particular network flow problem.

We introduce the algorithm by considering an example, and we then ask you to solve a problem for yourself. You may find it helpful to listen to the discussion of the example more than once before tackling the problem. If you still have difficulty with the problem, listen to the discussion of the solution on the tape, and then try to solve the problem for yourself in a later study session.

Both algorithms are designed for use on a computer. They are designed to be efficient for large examples. The examples we give here (to be solved without the use of a computer) are necessarily very simple, and have been chosen to illustrate the working of the algorithms and to demonstrate situations which may occur in practice. Because these examples are simple, it may be possible to solve them more quickly by other methods (for example, by inspection). You will be asked to use the algorithms to solve other examples in the computer activities for *Networks 1*.

3.3 The maximum flow algorithm

This tape sequence describes the maximum flow algorithm for finding a maximum flow and a minimum cut in a basic network. You should listen to this sequence and work through the accompanying frames at the relevant point in your study of *Networks 1*, Section 3.3.

A complete description of the algorithm is given after the tape frames.

Page 10 has been left blank to enable other frames to face each other.

4.1 The feasibility algorithm

In this tape sequence we give an algorithm for determining the feasibility of a given network with lower and upper capacities, and for finding a valid flow if the network proves to be feasible. You should listen to this sequence and work through the accompanying frames at the relevant point in your study of *Networks 1*, Section 4.1.

BLOCK 2

Notes for Graphs 2

There is one sequence on the tape for this unit. The heading number 4.2 below refers to the corresponding section in *Graphs 2*.

4.2 Multi-terminal flows

This tape sequence describes the algorithm of Gomory and Hu for obtaining a cut tree, and hence the maximum flow between each pair of vertices, in an *undirected* network.

We first describe an elementary form of the algorithm, and then the full algorithm, which includes the technique of condensing vertices.

Page 32 has been left blank to enable other frames to face each other.

Notes for Networks 2

There are two sequences on the tape for this unit. The heading number 2.1 below refers to the corresponding section in *Networks 2*.

Each sequence contains a worked problem, followed by a problem for you to do. Solutions to the problems are given in the tape frames.

2.1 Shortest and longest paths

The shortest path algorithm

In finding the shortest path from the source S to the sink T in a directed network, we assign a temporary label to each vertex, representing the shortest distance from S to that vertex by all paths so far considered. Eventually each vertex acquires a permanent label, called its *potential*, representing the shortest distance from S to that vertex. When T has been assigned a potential, the shortest distance from S to T has been found, and we can find the shortest path(s) from S to T by tracing back from T to S using the potentials.

Page 38 has been left blank to enable other frames to face each other.

The longest path algorithm

The algorithm is similar to the shortest path algorithm. However, it applies only to a network which has *no cycles;* otherwise we could go round the cycle an infinite number of times, and the procedure would not terminate.

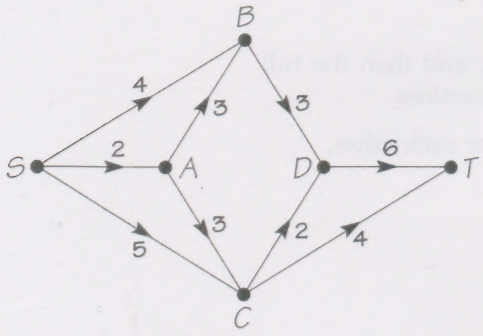
Page 44 has been left blank to enable other frames to face each other.

The maximum flow algorithm

TRACK 1

1 WORKED PROBLEM

Find a maximum flow and a minimum cut in the following basic network



USE
- labelling procedure
- flow-augmenting procedure

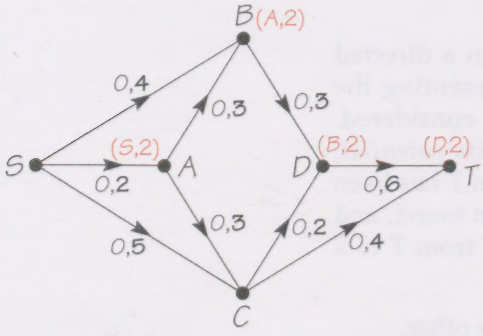
TRACK 2

2 SOLUTION TO WORKED PROBLEM

PART A: LABEL VERTICES

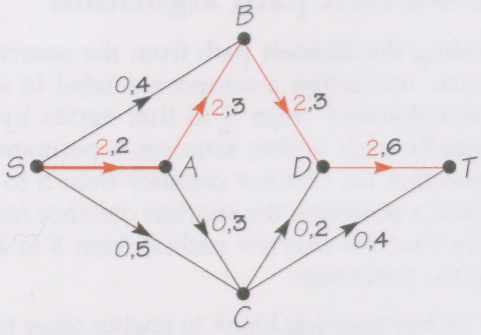
First iteration

Start with zero flow.



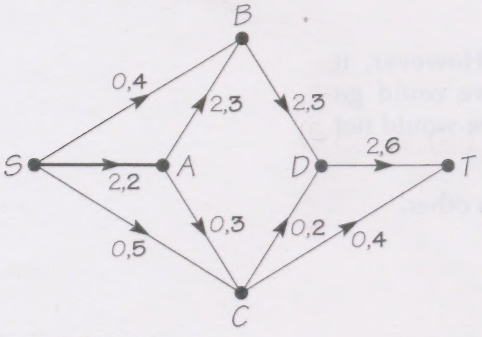
Label A, B, D, T.

PART B: AUGMENT FLOW

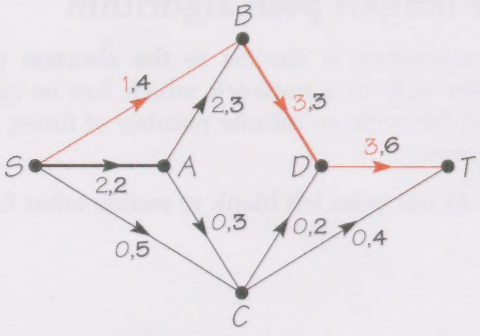


Increase flow along SABDT by 2.

Second iteration



Label B, D, T.



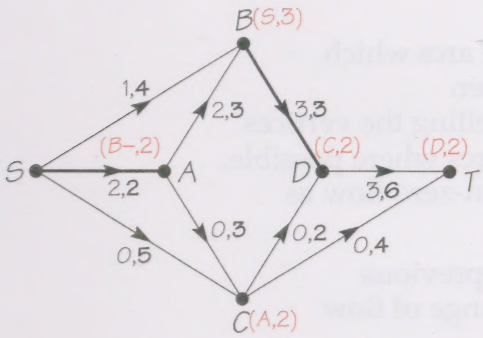
Increase flow along SBDT by 1.

TRACK 3

2 SOLUTION CONTINUED

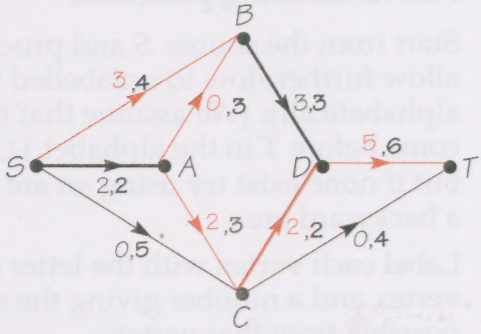
PART A: LABEL VERTICES

Third iteration



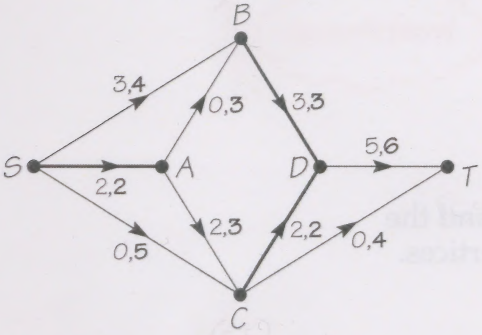
Label *B, A, C, D, T.*

PART B: AUGMENT FLOW

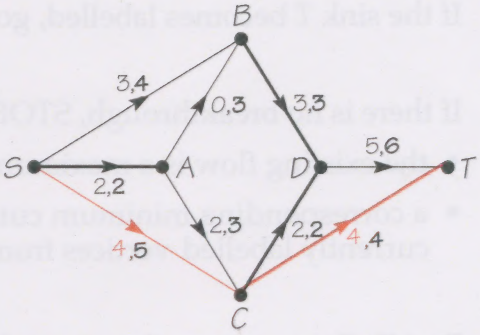


Increase flow along *SBACDT* by 2.

Fourth iteration

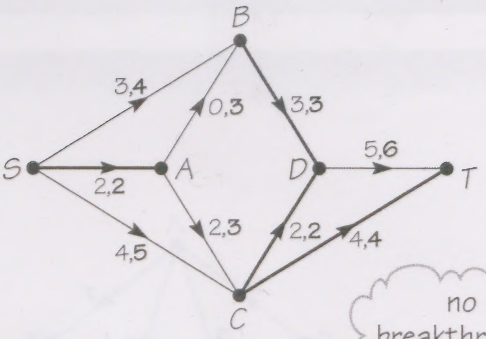


Label ~~*B*~~, *C, T.*



Increase flow along *SCT* by 4.

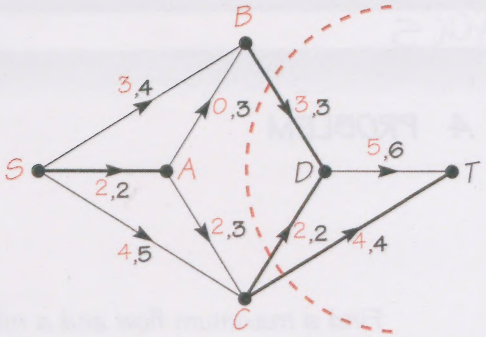
Fifth iteration



no breakthrough

Label *B, C, A.*

No more flow-augmenting paths.



Maximum flow: 5 + 4 = 9 = 2 + 1 + 2 + 4.

Minimum cut {*S, A, B, C*}, {*D, T*}

has capacity 3 + 2 + 4 = 9.

TRACK 4

3 SUMMARY OF THE ALGORITHM

START with any flow (possibly the zero flow).

Part A: labelling procedure

Start from the source S and proceed along the arcs which allow further flow to unlabelled vertices, chosen alphabetically. (We assume that the letters labelling the vertices come before T in the alphabet.) Use forward arcs where possible, but if none exist try using an arc carrying a non-zero flow as a backward arc.

Label each vertex with the letter denoting the previous vertex and a number giving the size of the change of flow possible from that vertex.

If a vertex is reached from which no further progress is possible, cross through the current label of that vertex and try backtracking to the vertex previously labelled, and continue from there.

If the sink T becomes labelled, go to Part B.

breakthrough

If there is no breakthrough, STOP:

- the existing flow is a maximum flow;
- a corresponding minimum cut separates S and the currently labelled vertices from the other vertices.

Part B: flow-augmenting procedure

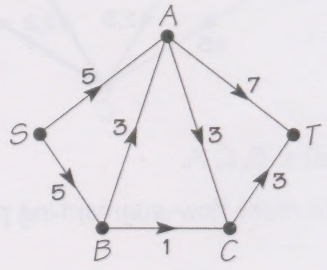
Augment the flow along the path of the labelled vertices $S, \ldots, T$ by the largest flow consistent with the labels.

Remove the labels and return to Part A.

TRACK 5

4 PROBLEM

Find a maximum flow and a minimum cut in the following basic network.



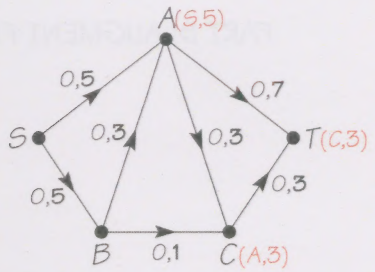
TRACK6

5 SOLUTION

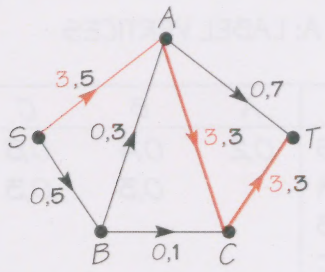
PART A: LABEL VERTICES

PART B: AUGMENT FLOW

1

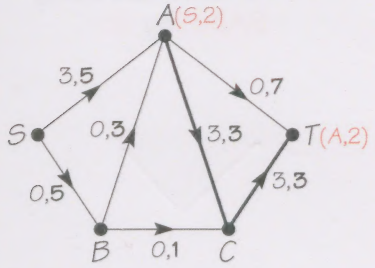


Label A, C, T.

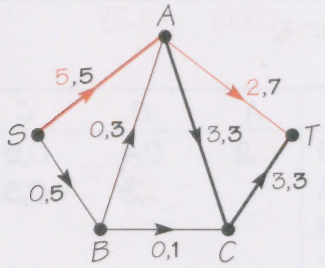


Increase flow along SACT by 3.

2

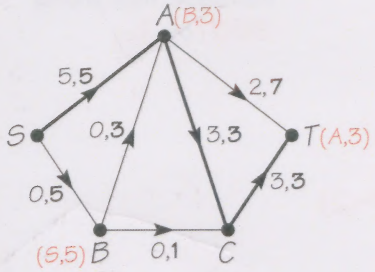


Label A, T.

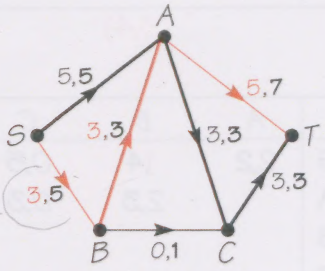


Increase flow along SAT by 2.

3

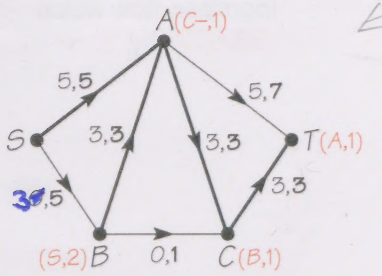


Label B, A, T.

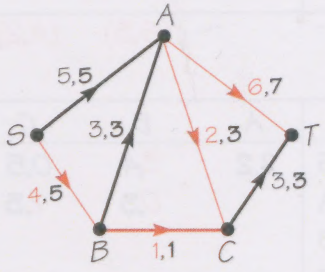


Increase flow along SBAT by 3.

4

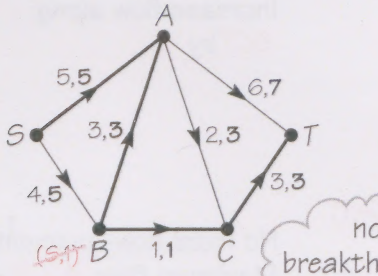


Label B, C, A, T.



Increase flow along SBCAT by 1.

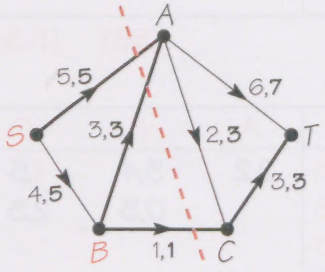
5



no breakthrough

Label B.

No more flow-augmenting paths.



Maximum flow: 6 + 3 = 9 = 3 + 2 + 3 + 1.

Minimum cut {S, B}, {A, C, T}

has capacity 5 + 3 + 1 = 9.

TRACK 7

6 WORKED PROBLEM

Find a maximum flow and a minimum cut in the basic network in Frame 1.

PART A: LABEL VERTICES

PART B: AUGMENT FLOW

1

	A	B	C	D	T
S	0,2	0,4	0,5		
A		0,3	0,3		
B				0,3	
C				0,2	0,4
D					0,6
	(S,2)	(A,2)		(B,2)	(D,2)

Increase flow along SABDT by 2.

2

	A	B	C	D	T
S	2,2	0,4	0,5		
A		2,3	0,3		
B				2,3	
C				0,2	0,4
D					2,6
		(S,4)		(B,1)	(D,1)

Increase flow along SBDT by 1.

3

	A	B	C	D	T	
S	2,2	1,4	0,5			
A		2,3	0,3			(B–,2)
B				3,3		
C				0,2	0,4	
D					3,6	
		(S,3)	(A,2)	(C,2)	(D,2)	

Increase flow along SBACDT by 2.

4

	A	B	C	D	T
S	2,2	3,4	0,5		
A		0,3	2,3		
B				3,3	
C				2,2	0,4
D					5,6
		~~(S,1)~~	(S,5)		(C,4)

Increase flow along SCT by 4.

5

	A	B	C	D	T	
S	2,2	3,4	4,5			
A		0,3	2,3			~~(C–,1)~~
B				3,3		
C				2,2	4,4	
D					5,6	
		~~(S,1)~~	(S,1)			

No more flow-augmenting paths.
Maximum flow: 4 + 5 = 9.
Minimum cut {S, A, B, C}, {D, T} has capacity 3 + 2 + 4 = 9.

TRACK 8

7 PROBLEM

Find a maximum flow and a minimum cut in the basic network in Frame 4.

PART A: LABEL VERTICES

PART B: AUGMENT FLOW

1

	A	B	C	T
S	0,5	0,5		
A			0,3	0,7
B	0,3		0,1	
C				0,3

Increase flow along
..... by .....

2

	A	B	C	T
S				
A				
B				
C				

Increase flow along
..... by .....

3

	A	B	C	T
S				
A				
B				
C				

Increase flow along
..... by .....

4

	A	B	C	T
S				
A				
B				
C				

Increase flow along
..... by .....

5

	A	B	C	T
S				
A				
B				
C				

No more flow-augmenting paths.
Maximum flow: .....
Minimum cut .....
has capacity .....

8 SOLUTION TO PROBLEM USING TABULAR METHOD

PART A: LABEL VERTICES

PART B: AUGMENT FLOW

1

	A	B	C	T
S	0,5	0,5		
A			0,3	0,7
B	0,3		0,1	
C				0,3
	(S,5)		(A,3)	(C,3)

Increase flow along SACT by 3.

2

	A	B	C	T
S	3,5	0,5		
A			3,3	0,7
B	0,3		0,1	
C				3,3
	(S,2)			(A,2)

Increase flow along SAT by 2.

3

	A	B	C	T
S	5,5	0,5		
A			3,3	2,7
B	0,3		0,1	
C				3,3
	(B,3)	(S,5)		(A,3)

Increase flow along SBAT by 3.

4

	A	B	C	T	
S	5,5	3,5			
A			3,3	5,7	(C−,1)
B	3,3		0,1		
C				3,3	
		(S,2)	(B,1)	(A,1)	

Increase flow along SBCAT by 1.

5

	A	B	C	T
S	5,5	4,5		
A			2,3	6,7
B	3,3		1,1	
C				3,3
		(S,1)		

No more flow-augmenting paths.
Maximum flow: 6 + 3 = 9.
Minimum cut $\{S,B\}, \{A,C,T\}$ has capacity 5 + 3 + 1 = 9.

Maximum flow algorithm

Part A: labelling procedure

START with a flow found by inspection, and with all the vertices unlabelled.

STEP 1 Choose an unsaturated arc SV_1 and assign to the vertex V_1 the label (S, a_1), where a_1 is the amount of flow needed to saturate the arc SV_1.

STEP 2 Choose an unlabelled vertex V_2 such that

either (a) V_1V_2 is an unsaturated arc
or (b) V_2V_1 is an arc with a non-zero flow.

In case (a), assign to the vertex V_2 the label (V_1, a), where a is either a_2, the amount needed to saturate V_1V_2, or a_1, whichever is the smaller.

In case (b), assign to the vertex V_2 the label (V_1^-, a), where a is either a_2, the amount of flow in V_2V_1, or a_1, whichever is the smaller.

Continue in this way, choosing vertices V_3, V_4, ...

At each stage, choose the vertex V_{k+1} to be any unlabelled vertex such that

either (a) V_kV_{k+1} is an unsaturated arc
or (b) $V_{k+1}V_k$ is an arc with a non-zero flow.

In case (a), assign to the vertex V_{k+1} the label (V_k, a), where a is either a_{k+1}, the amount needed to saturate V_kV_{k+1}, or a_k, whichever is the smaller.

In case (b), assign to the vertex V_{k+1} the label (V_k^-, a), where a is either a_{k+1}, the amount of flow in $V_{k+1}V_k$, or a_k, whichever is the smaller.

STEP 3 Continue in this way until

either (1) the vertex T becomes labelled (breakthrough)
or (2) there is no vertex V_{k+1} with property (a) or (b).

In case (1), a flow-augmenting path has been found, so proceed to Part B.

In case (2), backtrack along the path, crossing through the labels of vertices which do not lie on a flow-augmenting path, until vertex V_i is reached with the property that either there is an unsaturated arc V_iW or there is an arc WV_i carrying a non-zero flow, where W is as yet unlabelled. If such a vertex V_i is found, regard W as V_{i+1} and label it as in Step 2.

STEP 4 Continue in this way, labelling vertices where possible and backtracking where necessary until

either (a) vertex T becomes labelled (breakthrough)
or (b) no further labelling is possible.

In case (a), proceed directly to Part B.

In case (b), no increase in flow is possible, and a maximum flow has already been obtained. The arcs separating the source S and the labelled vertices from the other vertices form a minimum cut. STOP.

In the audio-tape and computer activities, we proceed systematically — considering vertices in alphabetical order, and choosing forward arcs rather than backward arcs where possible. However, these are not essential features of the algorithm. When working on small networks without the use of a computer, it is usually quicker to use the general version of the algorithm given here.

Part B: flow-augmenting procedure

The following procedure is used when the sink T has been labelled.

STEP 5 Look at the label of T. It will be of the form (V_k, a), for some vertex V_k and some positive number a. Increase the flow along V_kT by a.

STEP 6 Look at the label of V_k. It will be of the form (V_{k-1}, a) or $(V_{k-1}-, a)$ for some vertex V_{k-1}. In the former case, increase the flow along $V_{k-1}V_k$ by a, and in the latter case decrease the flow along V_kV_{k-1} by a.

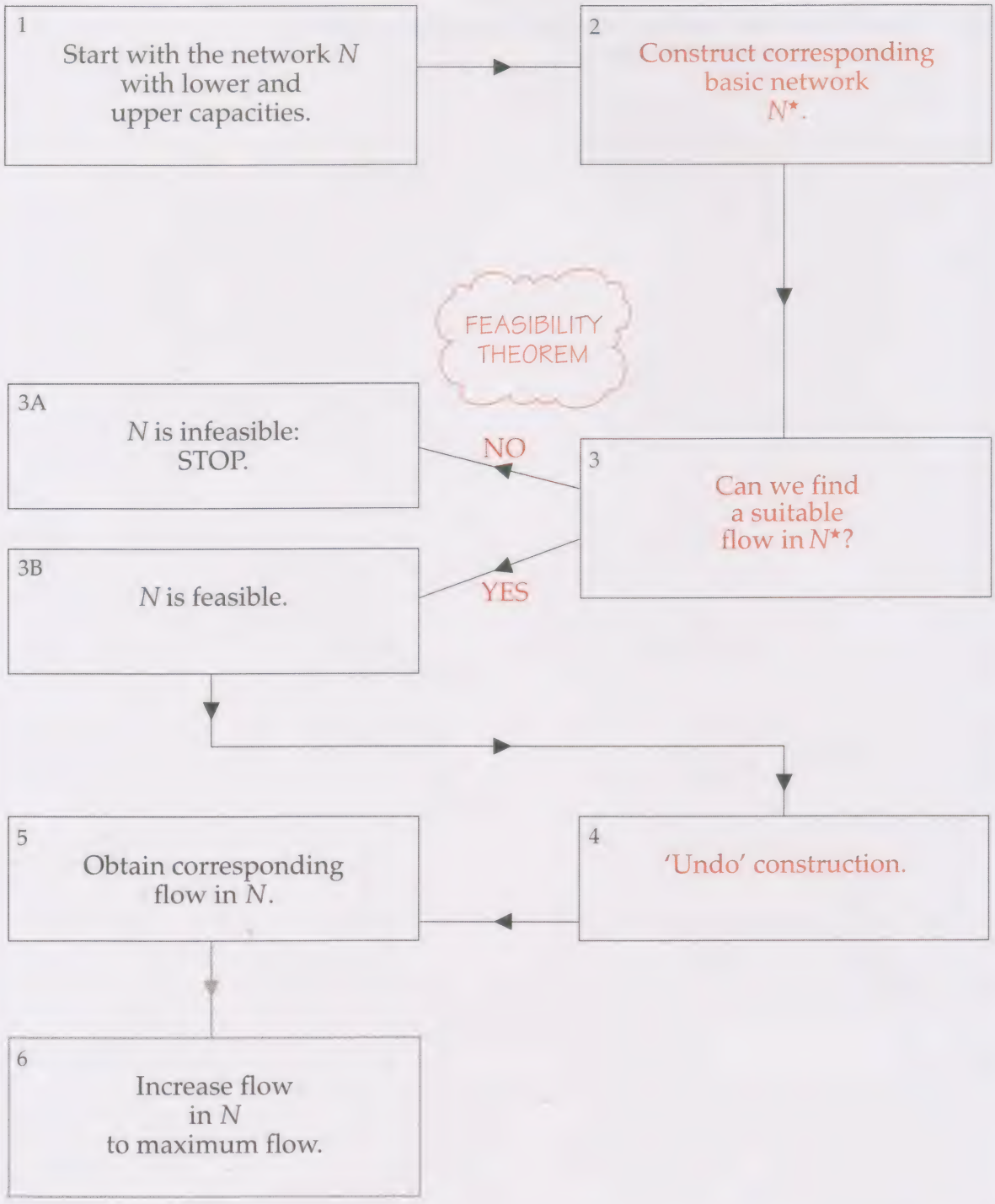
STEP 7 Repeat Step 7 successively for the vertices V_{k-1}, V_{k-2}, ..., V_1, and conclude by increasing the flow along SV_1 by a.

STEP 8 Erase all the labels, and start the labelling procedure (Step 1) using the flow obtained in Steps 5–7.

The feasibility algorithm

TRACK 9

1 OUTLINE OF THE ALGORITHM



2 CONSTRUCTION OF N*

Network N ➡ Network N^*

STAGE 1: take vertices of N^* to be vertices of N together with a new source S^* and a new sink T^*.

STAGE 2: for each arc VW, replace the capacities l, u by $u - l$.

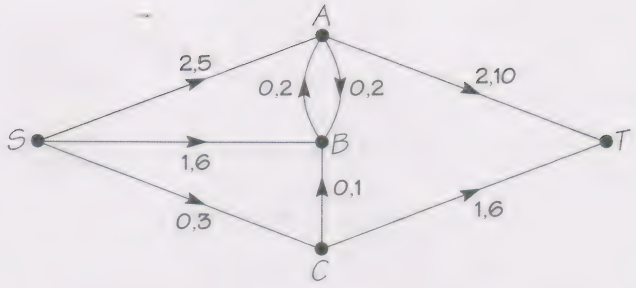
STAGE 3: for each arc VW with lower capacity l, add two new arcs: an arc S^*W and an arc VT^*, each with capacity l.

STAGE 4: add a new arc from T to S with infinite capacity.

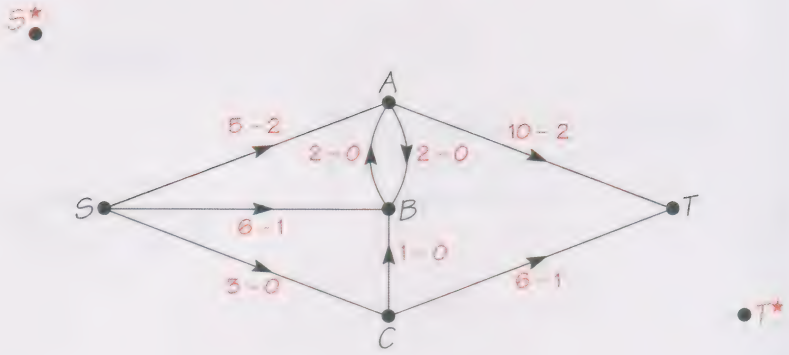
TRACK 11

3 EXAMPLE

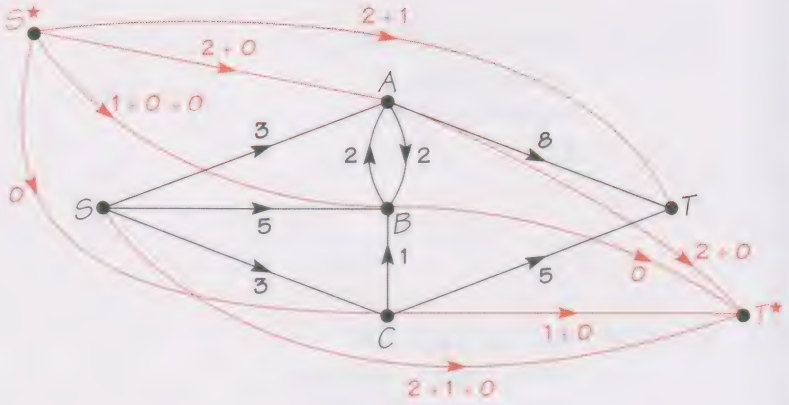
Consider the network N:



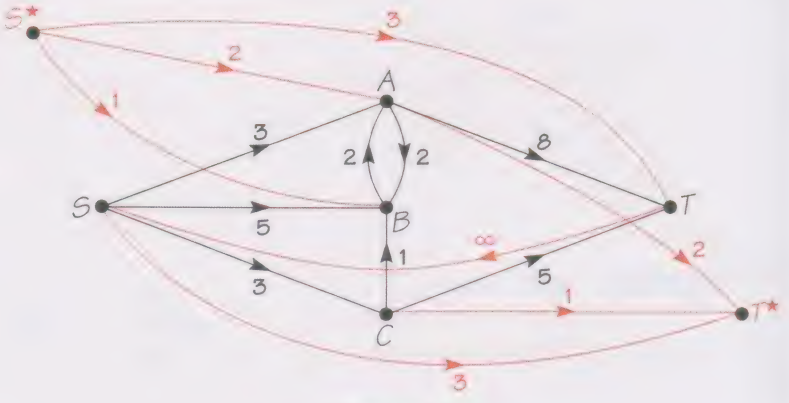
STAGES 1 and 2:
add $S^\star$ and $T^\star$ and replace capacities.



STAGE 3:
for each arc VW, add arcs $S^\star W$ and $VT^\star$.



STAGE 4:
add arc from T to S.

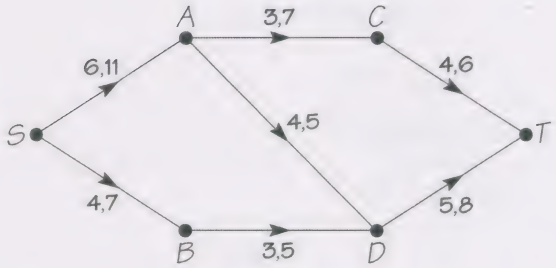


the network $N^\star$

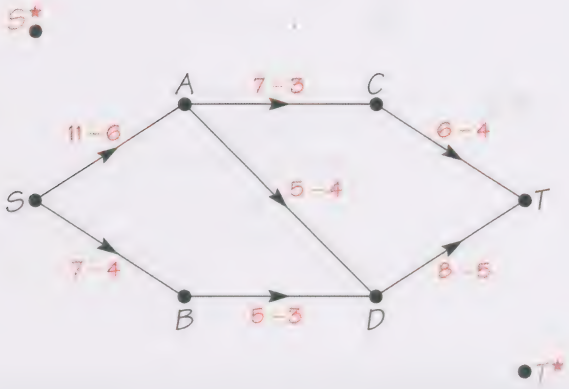
TRACK 12

4 PROBLEM

Consider the network N:

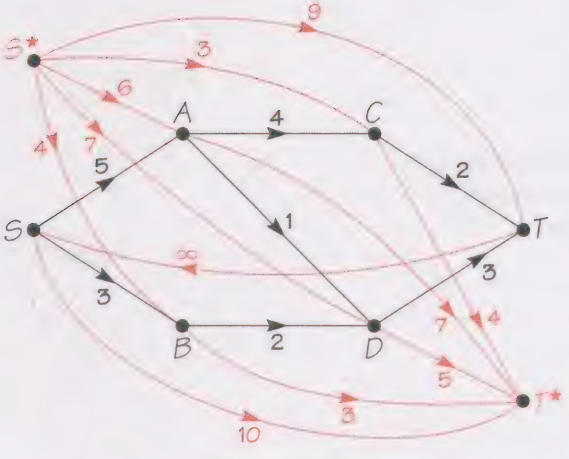


STAGES 1 and 2:



STAGE 3:

STAGE 4:

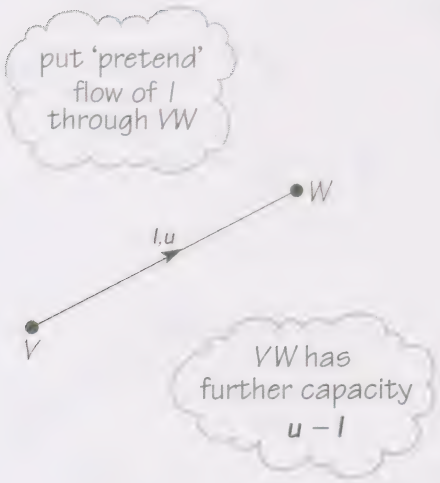


the network N^*

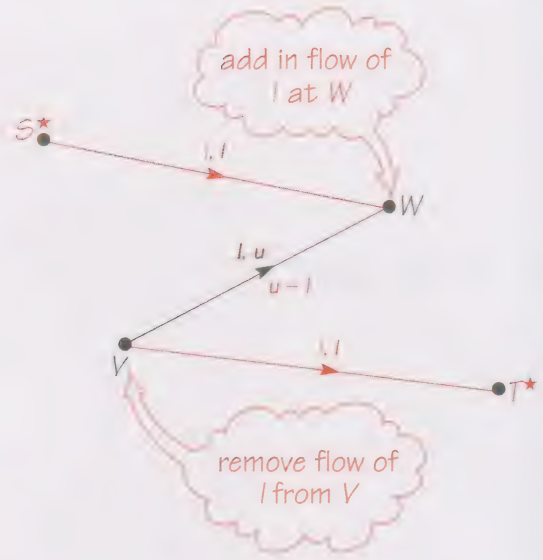
TRACK 13

5 UNDERLYING IDEA

typical arc in N



corresponding arc in N^*



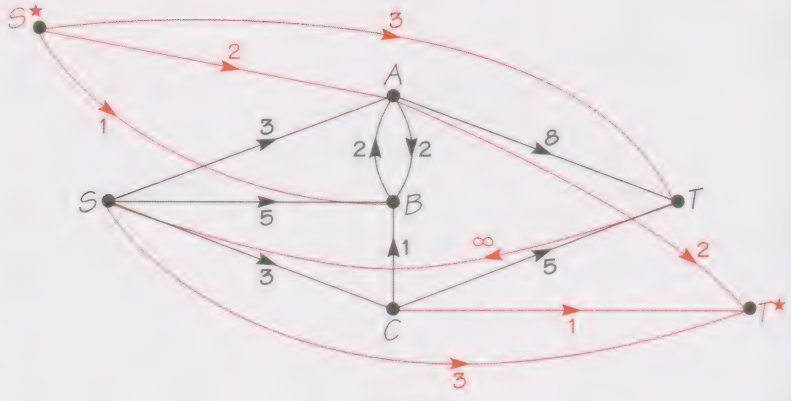
TRACK 14

6 FEASIBILITY THEOREM

a network N with lower and upper capacities is feasible $\iff$ in N^* there is a flow from S^* to T^* such that *all arcs out of S^* and all arcs into T^* are saturated*

7 EXAMPLE CONTINUED

Can we find a suitable flow in N^*?

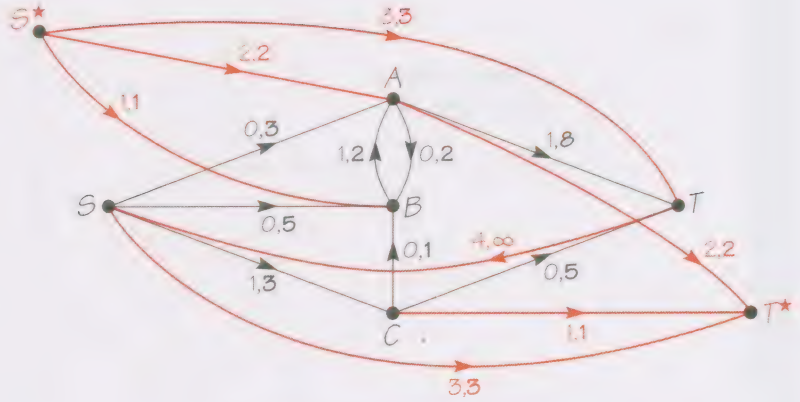


the network N^*

TRACK 15

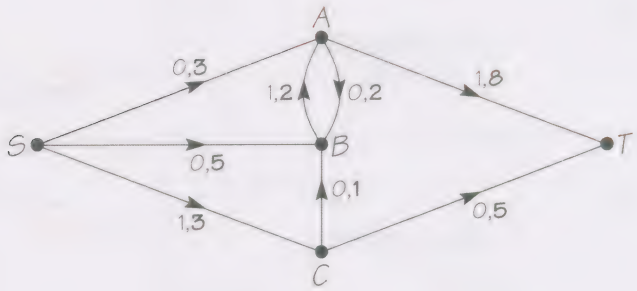
8 EXAMPLE CONTINUED

We can find a flow in N^* such that all arcs out of S^* and all arcs into T^* are saturated:
N is feasible.

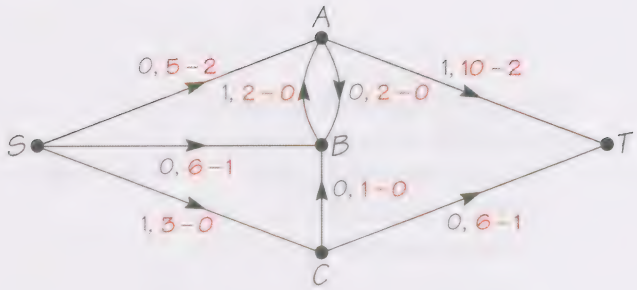


a suitable flow of value 6 in N^*

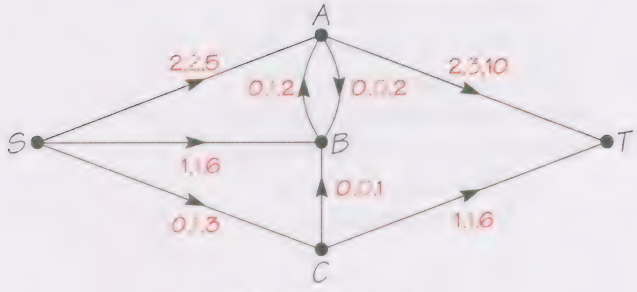
'Undo' stages 1, 3 and 4: remove S^* and T^*, and all extra arcs.



Next, we 'undo' stage 2.



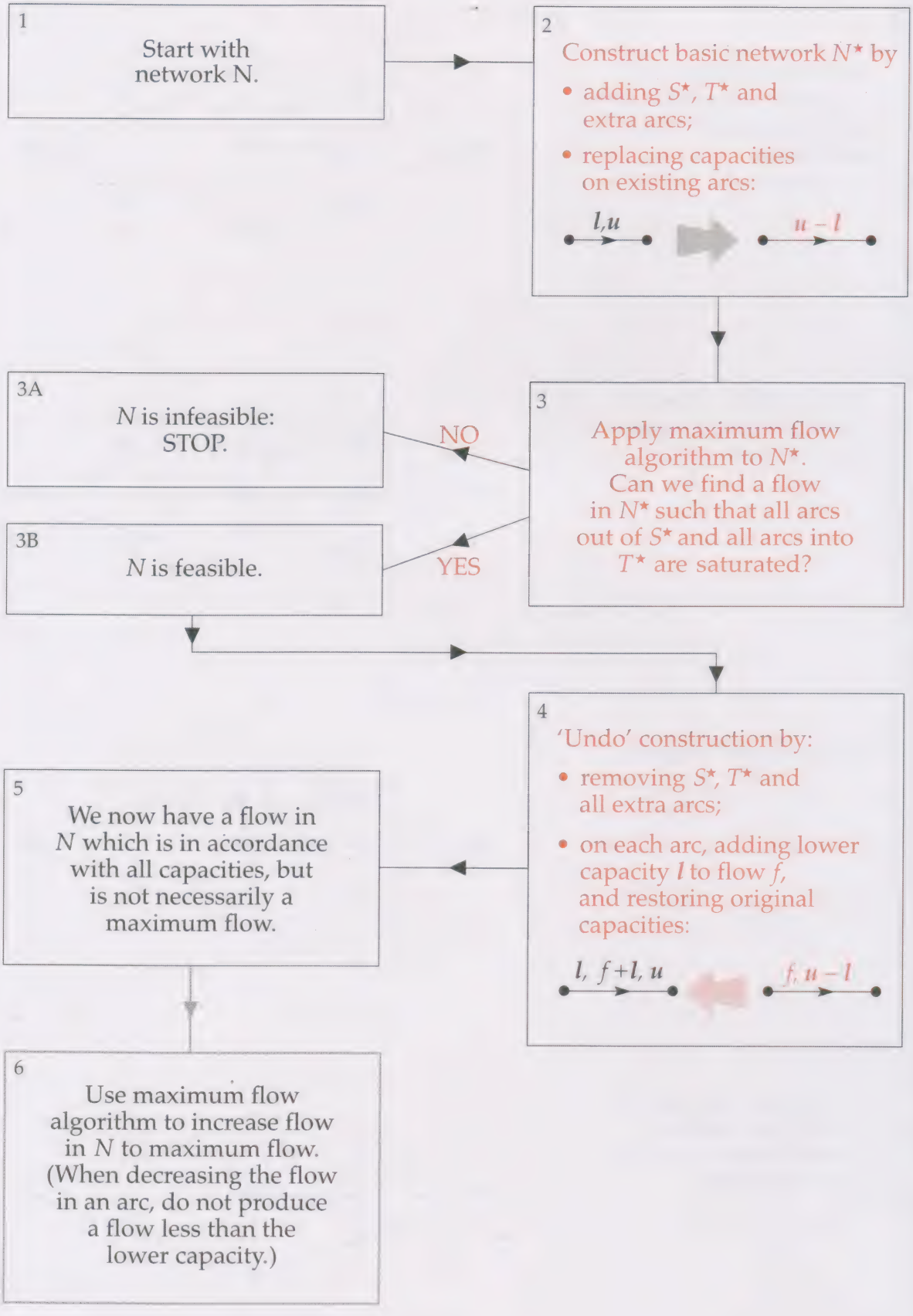
On each arc, add lower capacity to flow and restore original upper and lower capacities.



a flow of value 4 in N

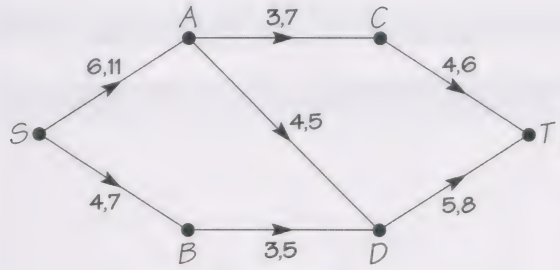
TRACK 16

9 THE ALGORITHM



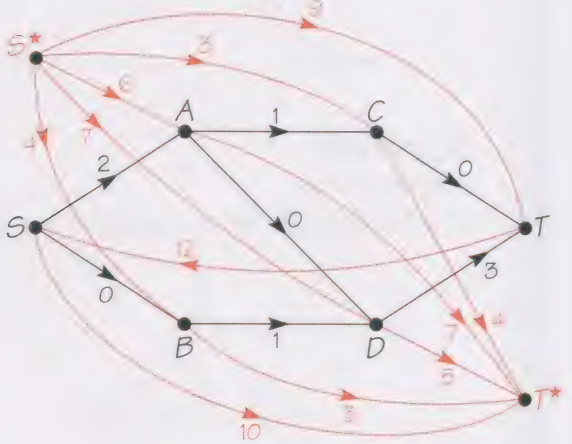
TRACK 16 & TRACK 17

10 PROBLEM CONTINUED



the network N

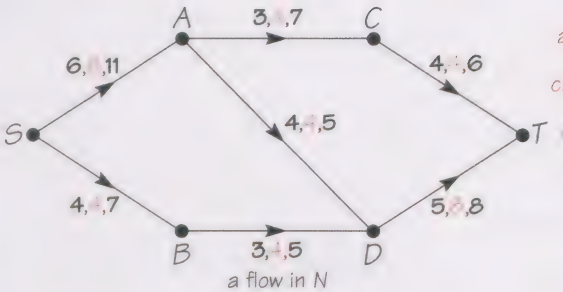
the basic network N*



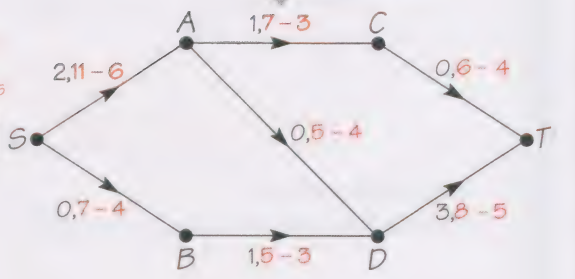
a suitable flow in N*

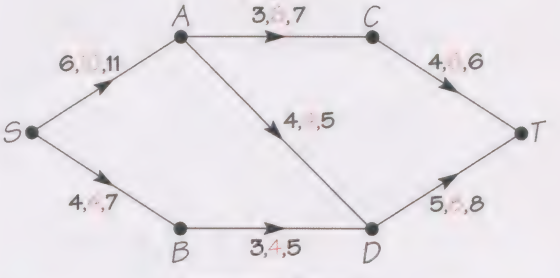
remove S*, T*, and red arcs

add back lower capacities



a flow in N





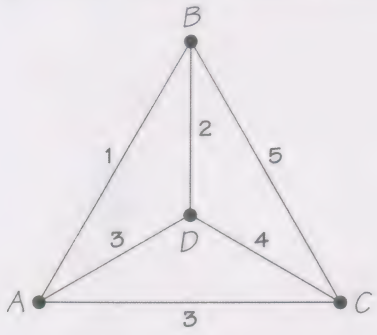
a maximum flow in N

The algorithm of Gomory and Hu

TRACK 18

1 WORKED PROBLEM 1

Find a cut tree for the following network.

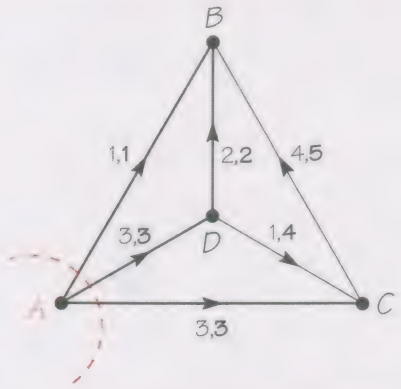


4 vertices, so 3 iterations

SOLUTION

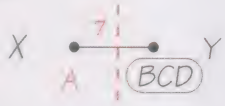
First iteration

Choose any two vertices, say A and B.



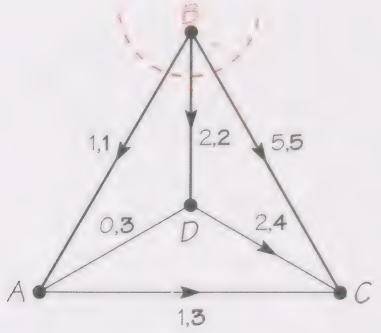
max flow from A to B

- max flow from A to B has value 7;
- min cut separates {A}, {B, C, D};
- first branch of tree:



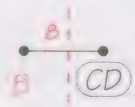
Second iteration

Choose any two vertices from (BCD), say B and C.

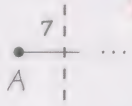


max flow from B to C

- max flow from B to C has value 8;
- min cut separates {B}, {A, C, D};
- new branch:



- join to:

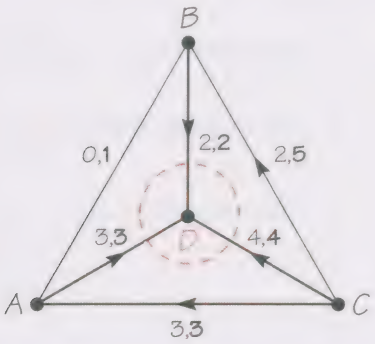


- result:

1 SOLUTION CONTINUED

Third iteration

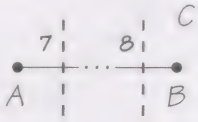
Consider the two vertices C and D.



max flow from C to D

- max flow from C to D has value 9;
- min cut separates {D}, {A, B, C};
- new branch:

- join to:



- cut tree:

TRACK 19

2 OUTLINE OF THE ALGORITHM

General iteration

choosing vertices
first iteration: choose any two;
later: choose two from any generalized vertex

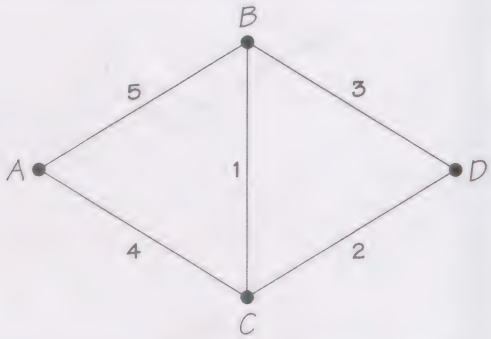
Choose two vertices.

To find the branch in the cut tree joining the two chosen vertices:

- find a maximum flow from the first vertex to the second, and find a corresponding minimum cut in the network;
- note how the minimum cuts splits all the vertices into two disjoint subsets;
- draw the corresponding new branch of the cut tree;
- draw the existing part(s) of the cut tree that the new branch must be joined to;
- join the new branch to the existing part of the cut tree.

3 PROBLEM 1

Use the algorithm to find a cut tree for the following network.
First find a maximum flow from A to B.

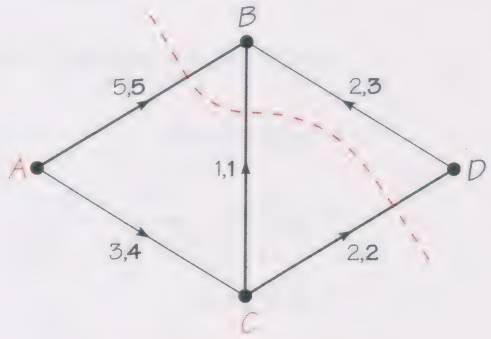


TRACK 20

4 SOLUTION TO PROBLEM 1

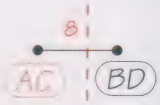
First iteration

Choose vertices A and B.



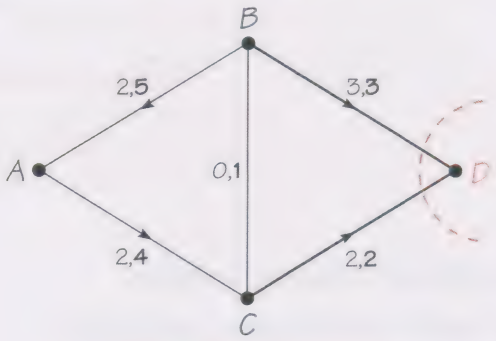
max flow from A to B

- max flow from A to B has value 8;
- min cut separates {A, C}, {B, D};
- first branch of tree:



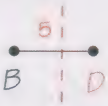
Second iteration

Split up generalized vertex (BD).

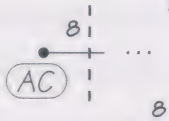


max flow from B to D

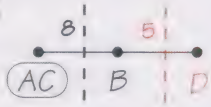
- max flow from B to D has value 5;
- min cut separates {D}, {A, B, C};
- new branch:



- join to:

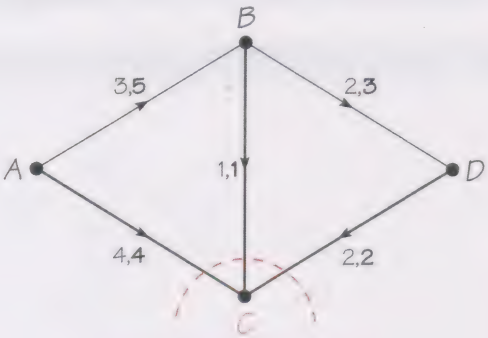


- result:



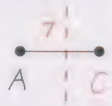
Third iteration

Split up generalized vertex (AC).

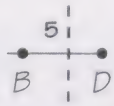


max flow from A to C

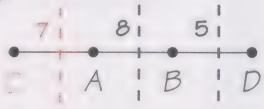
- max flow from A to C has value 7;
- min cut separates {C}, {A, B, D};
- new branch:



- join to:



- cut tree:

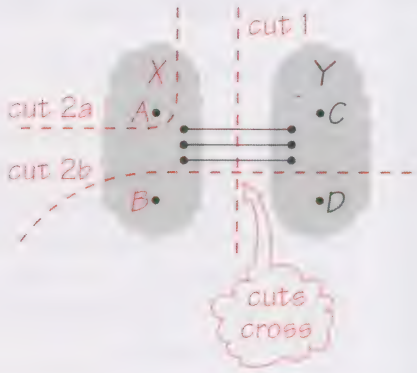


TRACK 21

5 DOES IT ALWAYS WORK?

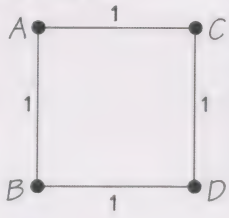
Suppose the first minimum cut (cut 1) separates the vertices into two disjoint sets X and Y.

Must the second minimum cut, separating vertices A and B in X, necessarily keep Y entirely on one side of the cut, so that Y may be joined to either A or B, as appropriate?

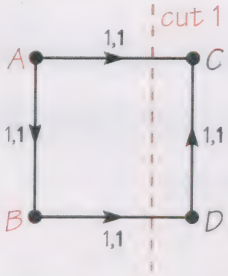


TRACK 22

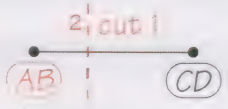
6 EXAMPLE



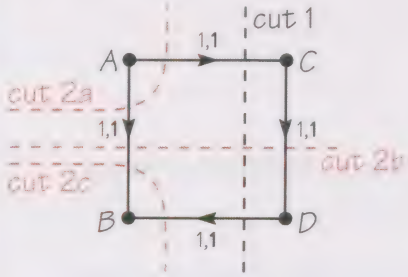
First iteration
Choose vertices A and C.



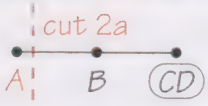
- max flow from A to C has value 2;
- min cut 1 separates $\{A, B\}$, $\{C, D\}$;
- first branch of tree:



Second iteration
Split up (AB).

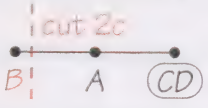


- max flow from A to B has value 2;
- min cut 2a separates $\{A\}$, $\{B, C, D\}$:



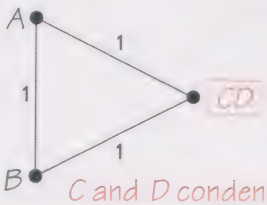
✓

- min cut 2b separates $\{A, C\}$, $\{B, D\}$; ✗
- min cut 2c separates $\{B\}$, $\{A, C, D\}$:

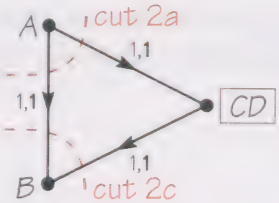


✓

We can avoid minimum cuts crossing by **condensing vertices**:



C and D condensed



- max flow from A to B still has value 2;
- min cut 2b is no longer possible;
- min cuts 2a and 2c remain.

TRACK 23

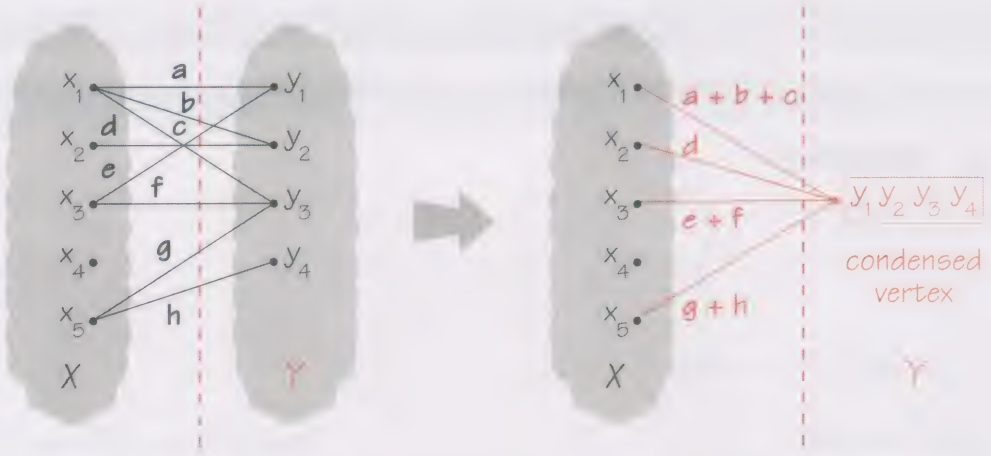
7 CONDENSING VERTICES

CONDENSING THEOREM

avoids cuts crossing

simplifies calculations

Consider a minimum cut in an undirected network N separating the vertices into two disjoint sets X and Y.
When finding a maximum flow between any two vertices in X, we may condense all the vertices in Y to a single vertex (and vice versa).
Edges connecting a vertex x in X to any vertex in Y are replaced by a single edge having capacity equal to the sum of the capacities of the connecting edges.



TRACK 24

8 OUTLINE OF FULL ALGORITHM OF GOMORY AND HU

General iteration

choosing vertices
first iteration: choose any two;
later: choose two from any generalized vertex

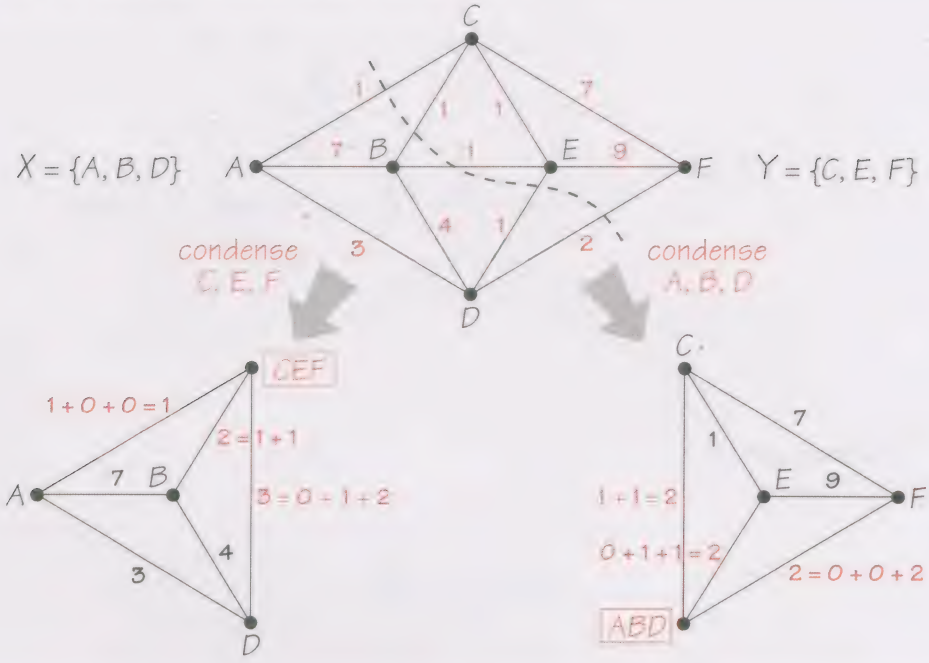
Choose two vertices from a generalized vertex.

To find a new branch of the cut tree:

- apply the condensing theorem appropriately;
- in the condensed network, find a maximum flow from the first vertex to the second, and find a corresponding minimum cut;
- note how the minimum cut splits all the vertices into two disjoint subsets;
- draw the corresponding new branch of the cut tree;
- join the new branch to the existing part of the cut tree so that the cut in the network corresponds to the cut in the tree.

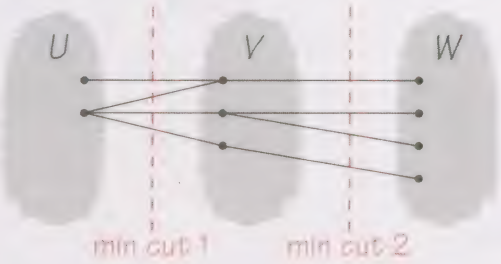
TRACK 25

9 EXAMPLE OF CONDENSING VERTICES



TRACK 26

10 USING THE CONDENSING THEOREM TWICE



When finding a maximum flow between 2 vertices in V, we may:

- condense all the vertices in U (applying the theorem to min cut 1);
- condense all the vertices in W (applying the theorem to min cut 2).

TRACK 27

11 PROBLEM 2

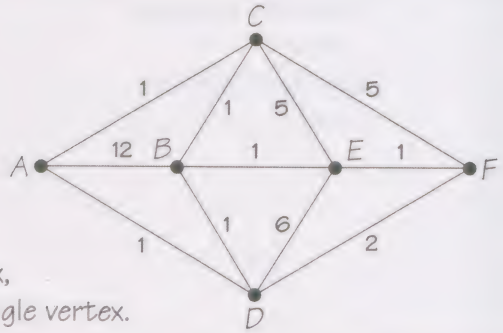
In the given network,

(a) condense the vertices C, D, E and F;

(b) condense the vertices A and B;

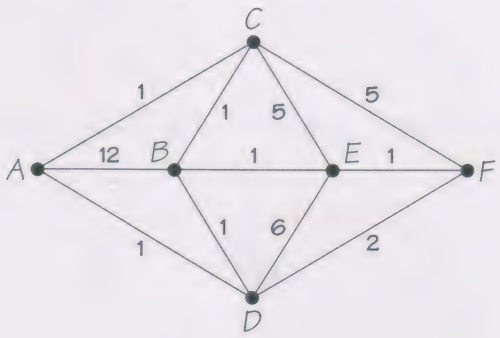
(c) condense the vertices A, B, D and E;

(d) condense the vertices A and B to a single vertex, and then condense the vertices C and F to a single vertex.

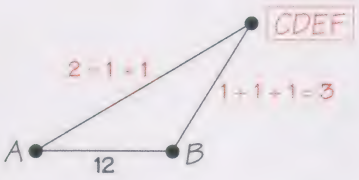


NO TRACK

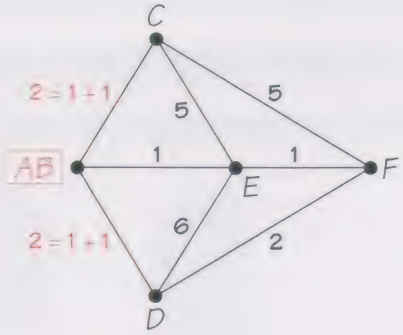
12 SOLUTION TO PROBLEM 2



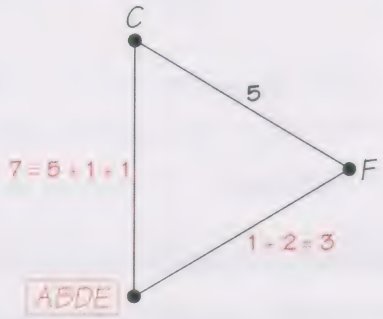
(a) Condense vertices C, D, E and F.



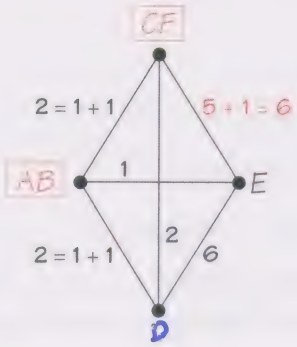
(b) Condense vertices A and B.



(c) Condense vertices A, B, D and E.



(d) Condense vertices A and B, and then vertices C and F.

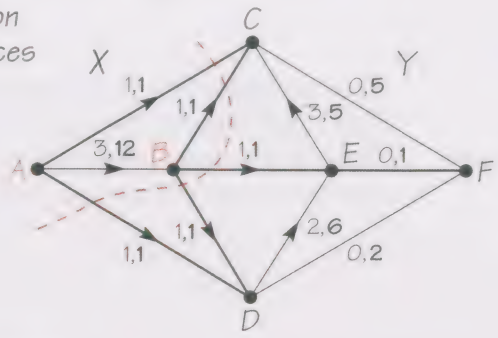


TRACK 28

13 WORKED PROBLEM 2

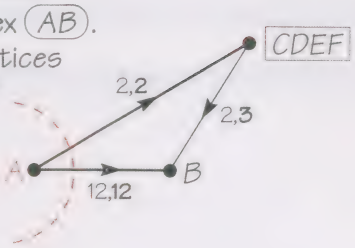
Find a cut tree for the network in Frame 11.

First iteration
Choose vertices A and C.



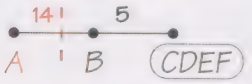
- max flow from A to C has value 5;
- min cut separates {A, B}, {C, D, E, F};
- first branch of tree:

Second iteration
Split up vertex (AB).
Condense vertices C, D, E and F

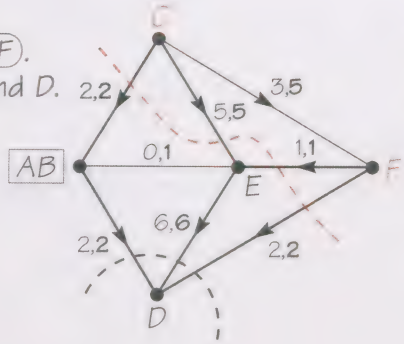


- max flow from A to B has value 14;
- min cut separates {A}, {B, C, D, E, F};
- new branch:

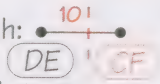
- join branch to tree:



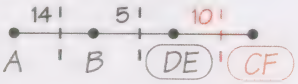
Third iteration
Split up vertex (CDEF).
Choose vertices C and D.
Condense vertices A and B.



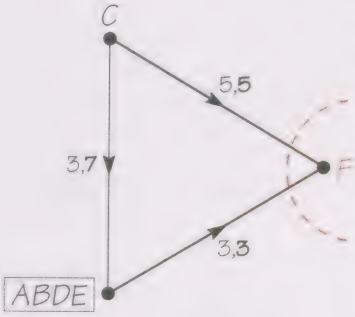
- max flow from C to D has value 10;
- min cut separates {C, F}, {A, B, D, E};
- new branch:



- join branch to tree:

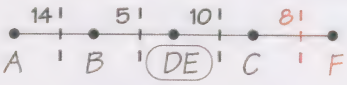


Fourth iteration
Split up vertex (CF).
Condense vertices A, B, D and E.

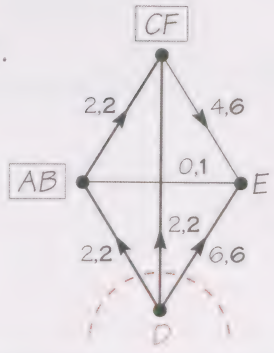


- max flow from C to F has value 8;
- min cut separates {F}, {A, B, C, D, E};
- new branch:

- join branch to tree:

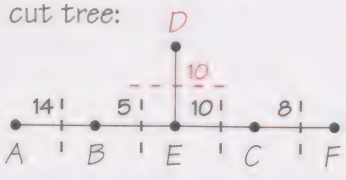


Fifth iteration
Split up vertex (DE).
Condense vertices A and B,
and then vertices C and F.



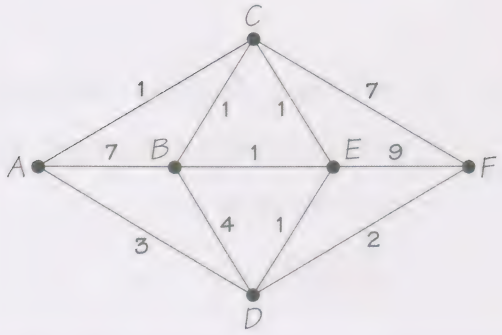
- max flow from D to E has value 10;
- min cut separates {D}, {A, B, C, E, F};
- new branch:

- cut tree:



TRACK 29

14 PROBLEM 3



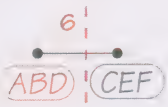
Use the algorithm of Gomory and Hu to find a cut tree for the given network.
First find a maximum flow from A to C.
Hint Use the condensed vertices in Frame 9.

TRACK 30

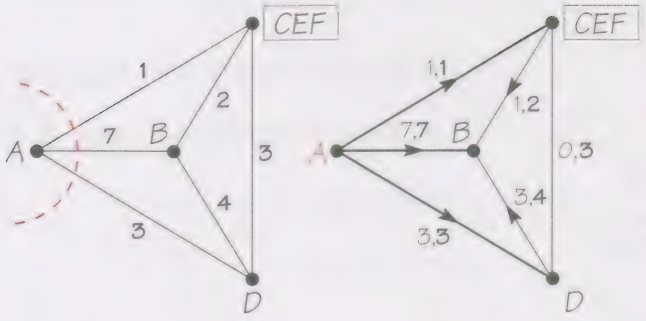
15 SOLUTION TO PROBLEM 3

First iteration
Choose vertices A and C.

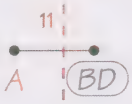
- max flow from A to C has value 6;
- min cut separates {A, B, D}, {C, E, F};
- first branch of tree:



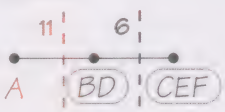
Second iteration
Split up vertex (ABD).
Choose vertices A and B.
Condense vertices C, E and F.



- max flow from A to B has value 11;
- min cut separates {A}, {B, C, D, E, F};
- new branch:

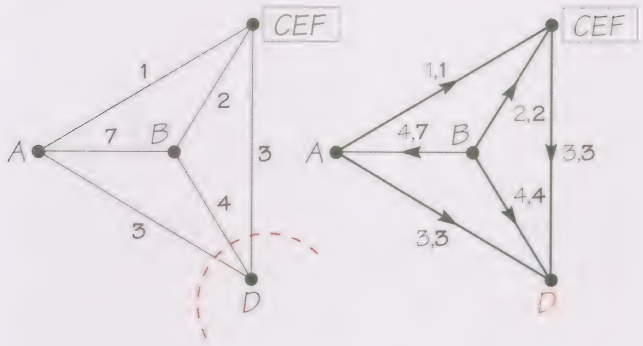


- join branch to tree:



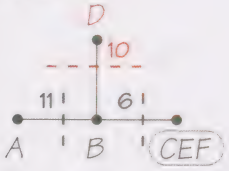
15 SOLUTION CONTINUED

Third iteration
Split up vertex (BD).
Condense vertices C, E and F.

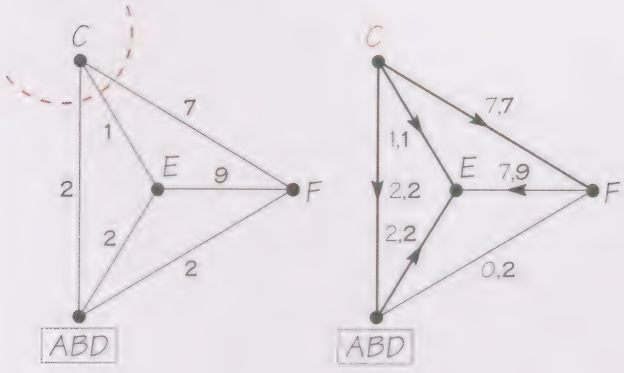


- max flow from B to D has value 10;
- min cut separates {D}, {A, B, C, E, F};
- new branch:

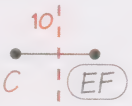
- join branch to tree:



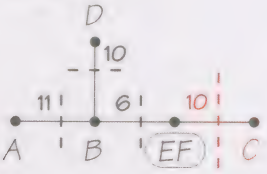
Fourth iteration
Split up vertex (CEF).
Choose vertices C and E.
Condense vertices A, B and D.



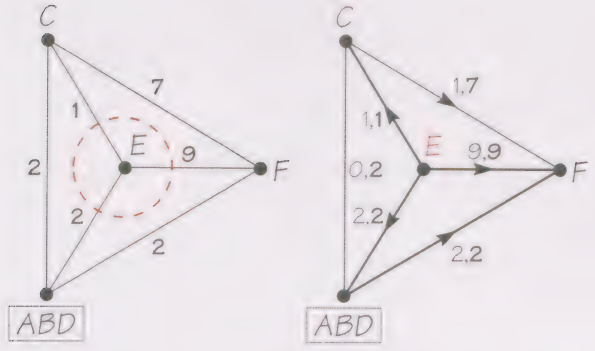
- max flow from C to E has value 10;
- min cut separates {C}, {A, B, D, E, F};
- new branch:



- join branch to tree:

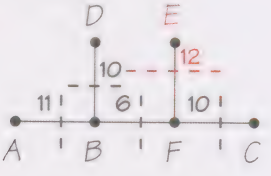


Fifth iteration
Split up vertex (EF).
Condense vertices A, B and D.



- max flow from E to F has value 12;
- min cut separates {E}, {A, B, C, D, F};
- new branch:

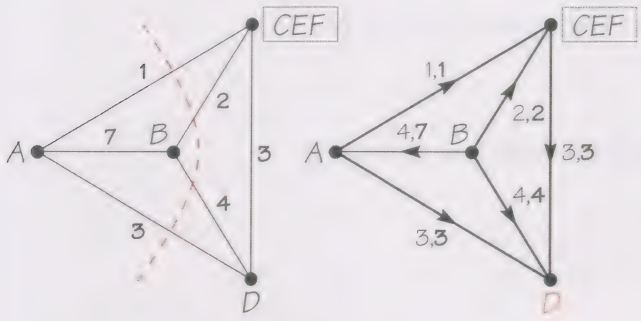
- join branch to tree:



TRACK 31

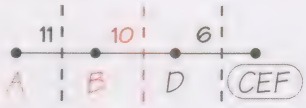
16 ANOTHER POSSIBLE SOLUTION TO PROBLEM 3

Third iteration
Split up vertex (BD).
Condense vertices C, E and F.

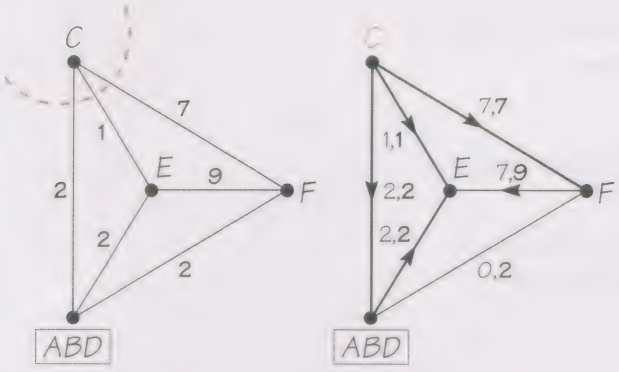


- max flow from B to D has value 10;
- min cut separates {A, B}, {C, D, E, F};
- new branch:

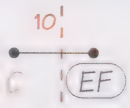
- join branch to tree:



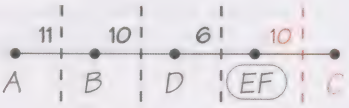
Fourth iteration
Split up vertex (CEF).
Choose vertices C and E.
Condense vertices A, B and D.



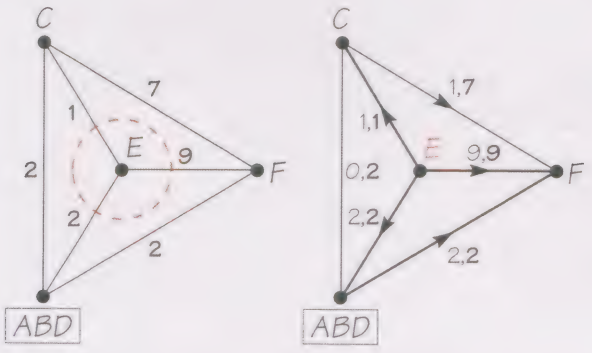
- max flow from C to E has value 10;
- min cut separates {C}, {A, B, D, E, F};
- new branch:



- join branch to tree:

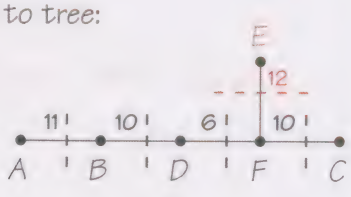


Fifth iteration
Split up vertex (EF).
Condense vertices A, B and D.



- max flow from E to F has value 12;
- min cut separates {E}, {A, B, C, D, F};
- new branch:

- join branch to tree:



Algorithm of Gomory and Hu for finding a cut tree

START with an undirected network N with n vertices.

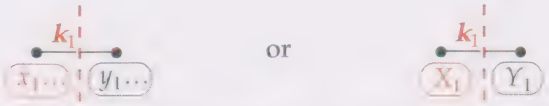
draw first branch

First iteration

Choose any two vertices, x_1 and y_1 say, in N.

- In N, find a maximum flow from x_1 to y_1 of value k_1, say.
- In N, find a minimum cut of capacity k_1 separating the vertices of N into two disjoint sets

  $X_1 = \{x_1, \ldots\}$, $Y_1 = \{y_1, \ldots\}$.
- Draw a branch with two vertices corresponding to the sets X_1 and Y_1, and with capacity k_1:



  At least one of the vertices, (X_1) say, will be a generalized vertex.

Second iteration

split up generalized vertex X_1

Choose any two vertices x_2 and x_3 in X_1.

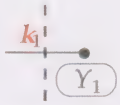
Condense all the vertices in Y_1 to a single vertex.

- In the condensed network, find a maximum flow from x_2 to x_3 of value k_2, say.
- In the condensed network, find a minimum cut of capacity k_2 separating the vertices of N into two disjoint sets

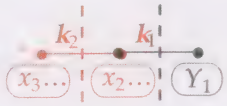
  $X_2 = \{x_2, \ldots\}$, $Y_2 = \{x_3, \ldots\}$. (*)
- The cut (*) splits the vertices of X_1 into two disjoint subsets $\{x_2, \ldots\}$ and $\{x_3, \ldots\}$.

  Draw a branch joining two vertices (possibly generalized) corresponding to these subsets, and with capacity k_2:

- Join the new branch to the existing part of the tree as follows:

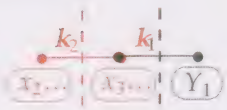


  at $(x_2\ldots)$ if $(x_2\ldots)$ belongs to the same part of the cut (*) as (Y_1):



cuts in tree must correspond to cuts in N

  at $(x_3\ldots)$ otherwise:



ONE OF THESE CASES MUST OCCUR.

General iteration

Split up a generalized vertex as follows.

- Choose two vertices from any generalized vertex.
- Condense the vertices in the given network as appropriate.
- In the condensed network,
  find a maximum flow from one of the chosen vertices to the other;
  find a corresponding minimum cut;
  draw the new branch of the cut tree;
  add the new branch to the existing part of the cut tree.

see Frame 7

in the general case there are so many possibilities that precise instructions cannot be given here — a study of the examples shows the methods required

Note that for a general iteration, the situation becomes more complicated because

- it may be possible to condense more than one set of vertices;
- the new branch may have to be joined to a vertex which is not an end vertex.

see Frames 10 and 13

see Frames 1 and 13

After n iterations, STOP: a cut tree has been obtained.

Points to note when applying the algorithm

- When splitting up a generalized vertex containing more than two vertices, it does not matter which two vertices you choose.
- There may be more than one way of achieving a maximum flow — choose any.
- For a particular maximum flow, there may be more than one minimum cut — choose any such that cuts *do not cross*.
- For a given undirected network, there may be more than one cut tree.

see Frames 5 and 6

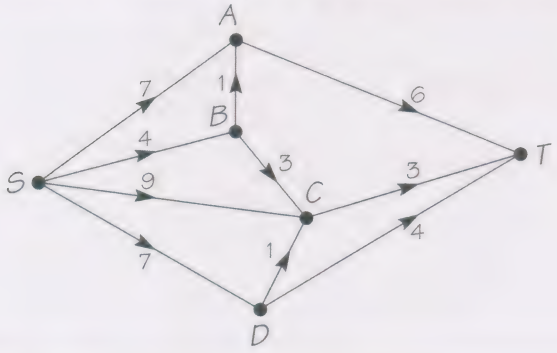
see Frame 16

TRACK 1

The shortest path algorithm

1 WORKED PROBLEM

Find the shortest path(s) from S to T in the given network.



USE

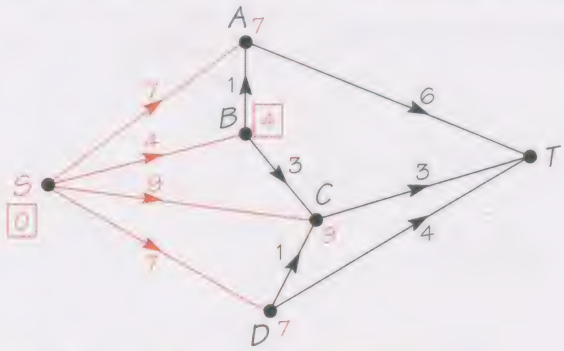
- labelling procedure
- tracing back procedure

TRACK 2 & TRACK 3

2 SOLUTION TO WORKED PROBLEM

Start by giving S zero potential.

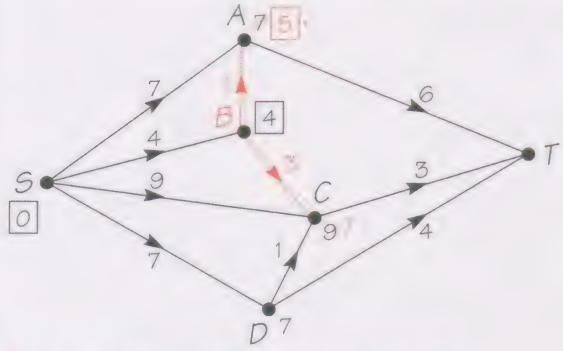
First iteration: label vertices reached directly from S.



iteration	origin vertex	vertices assigned labels				
		A	B	C	D	T
1	S	7	[4]	9	7	

B has potential 4 assigned from S.

Second iteration: label vertices reached directly from B.

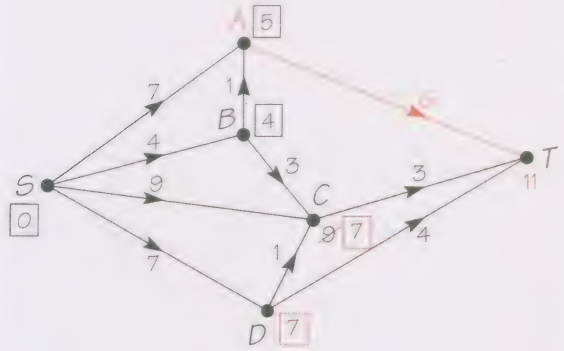


iteration	origin vertex	vertices assigned labels				
		A	B	C	D	T
1	S	7	[4]	9	7	
2	B	[5]		7		

A has potential 5 assigned from B.

2 SOLUTION CONTINUED

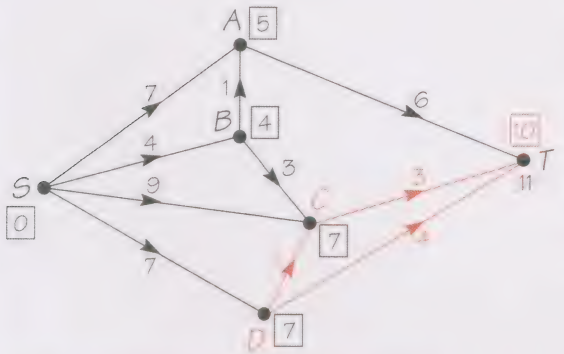
Third iteration: label vertices reached directly from A.



iteration	origin vertex	vertices assigned labels				
		A	B	C	D	T
1	S	~~7~~	4	~~9~~	7	
2	B	5		7		
3	A			7	7	11

C has potential 7 assigned from B.
D has potential 7 assigned from S.

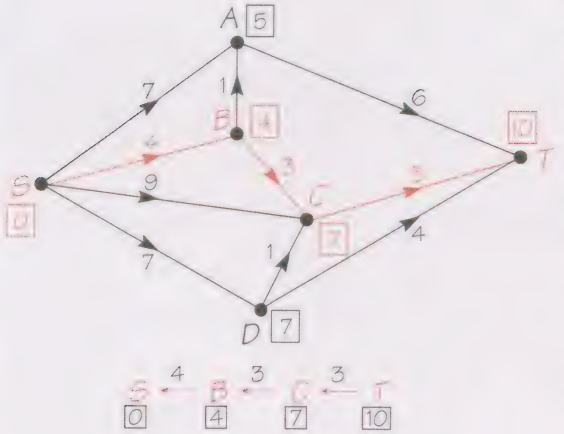
Fourth iteration: label vertices reached directly from C or D.



iteration	origin vertex	vertices assigned labels				
		A	B	C	D	T
1	S	~~7~~	4	~~9~~	7	
2	B	5		7		
3	A			7	7	11
4	C					10
	D					11

T has potential 10 assigned from C.

The shortest path



iteration	origin vertex	vertices assigned labels				
		A	B	C	D	T
1	S	~~7~~	4	~~9~~	7	
2	B	5		7		
3	A			7	7	11
4	C					10
	D					11

Shortest path is SBCT with length 10.

TRACK 4

3 THE SHORTEST PATH ALGORITHM: THE MAIN STEPS

START — Assign potential 0 to S.

GENERAL STEP — Consider the vertex (or vertices) just assigned a potential.

For each such vertex V, consider each vertex W which can be reached from V along an arc VW, and assign W the label

(potential of V) + (distance VW)

unless W already has a *smaller* label assigned from an earlier iteration.

When all such vertices W have been labelled, choose the smallest vertex label in the network which is not already a potential, and make it a potential at each vertex where it occurs.

REPEAT — the GENERAL STEP with the new potential(s).

STOP — when T has been assigned a potential.

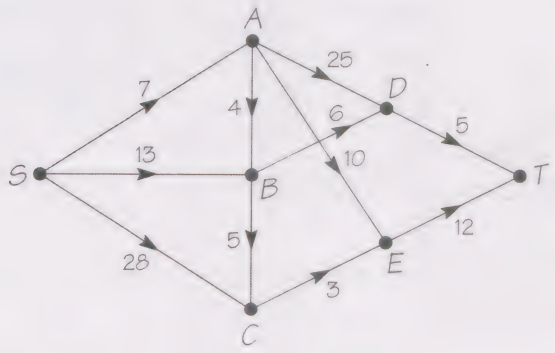
The shortest distance from S to T *is* the potential of T.
To find a shortest path, trace backwards from T and include an arc VW wherever

(potential of W) – (potential of V) = distance VW

until S is reached.

4 PROBLEM

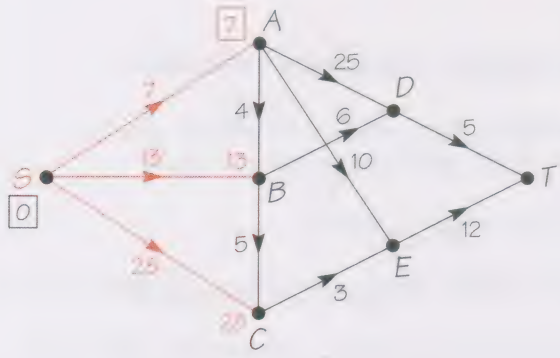
Find the shortest path from S to T in the following network.



5 SOLUTION TO PROBLEM

Start by giving S zero potential.

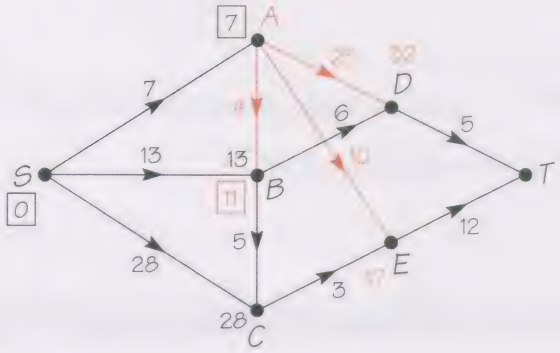
First iteration: label vertices reached directly from S.



iteration	origin vertex	vertices assigned labels A	B	C	D	E	T
1	S	7	13	28			

A has potential 7 assigned from S.

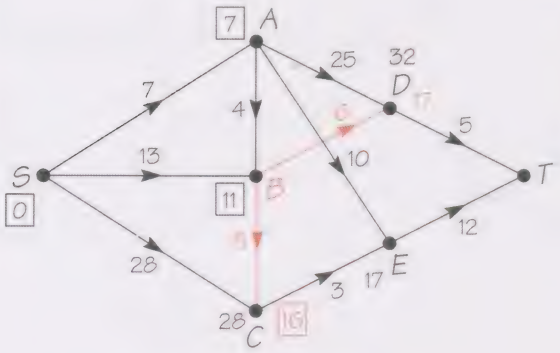
Second iteration: label vertices reached directly from A.



iteration	origin vertex	vertices assigned labels A	B	C	D	E	T
1	S	7	13	28			
2	A		11		32	17	

B has potential 11 assigned from A.

Third iteration: label vertices reached directly from B.

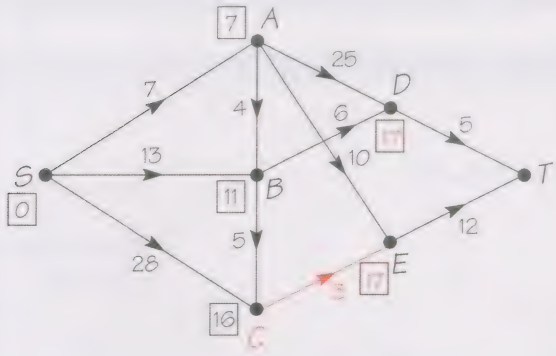


iteration	origin vertex	vertices assigned labels A	B	C	D	E	T
1	S	7	13	28			
2	A		11		32	17	
3	B			16	17		

C has potential 16 assigned from B.

5 SOLUTION CONTINUED

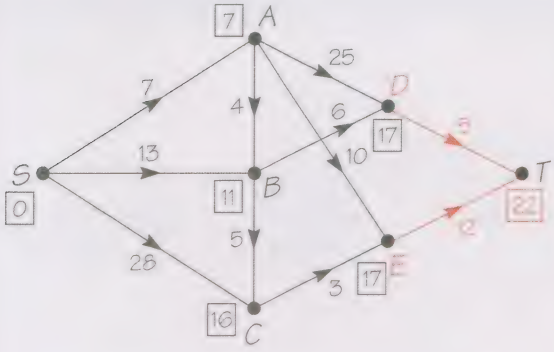
Fourth iteration: label vertices reached directly from *C*.



iteration	origin vertex	vertices assigned labels A	B	C	D	E	T
1	S	7	13	28			
2	A		11		32	17	
3	B			16	17		
4	C				17	17	

D has potential 17 assigned from *B*;
E has potential 17 assigned from *A*;

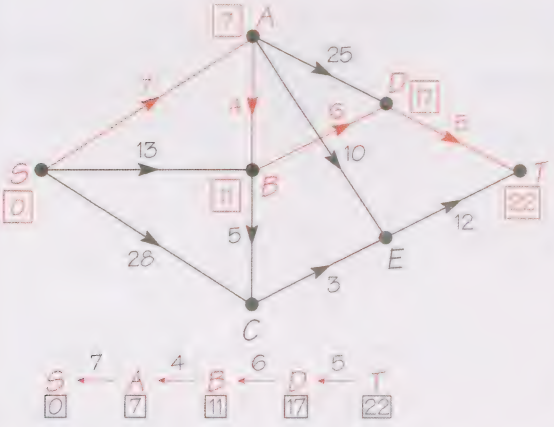
Fifth iteration: label vertices reached directly from *D* or *E*.



iteration	origin vertex	vertices assigned labels A	B	C	D	E	T
1	S	7	13	28			
2	A		11		32	17	
3	B			16	17		
4	C				17	17	
5	D						22
	E						29

T has potential 22 assigned from *D*.

The shortest path



iteration	origin vertex	vertices assigned labels A	B	C	D	E	T
1	S	7	13	28			
2	A		11		32	17	
3	B			16	17		
4	C				17	17	
5	D						22
	E						29

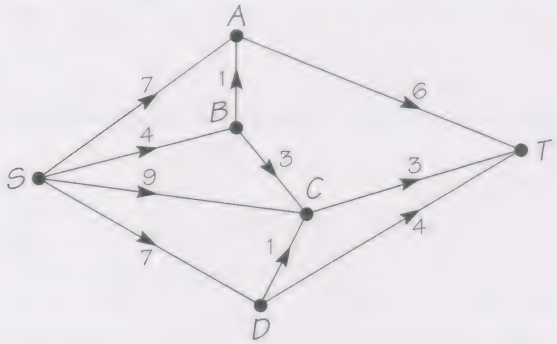
Shortest path is *SABDT* with length 22.

The longest path algorithm

TRACK 5

1 WORKED PROBLEM

Find the longest path(s) from S to T in the given network.



network with NO cycles

USE

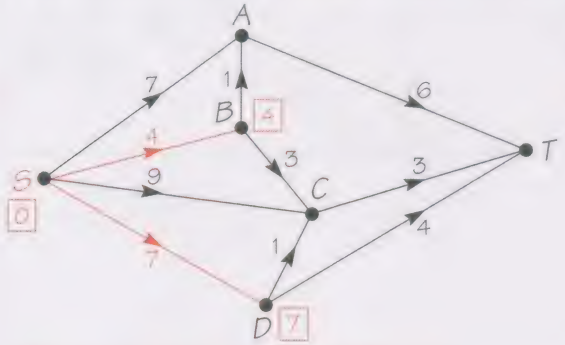
- labelling procedure
- tracing back procedure

TRACK 6 & TRACK 7

2 SOLUTION TO WORKED PROBLEM

Start by giving S zero potential.

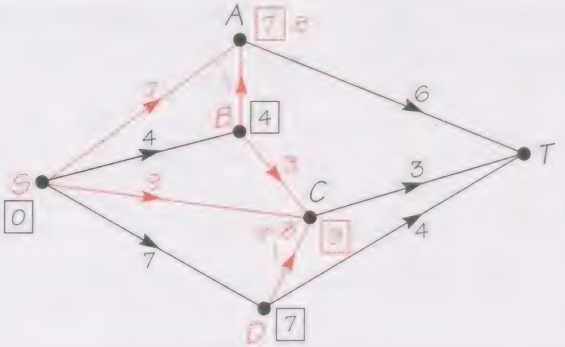
First iteration: label vertices reached directly only from S.



iteration	vertices assigned labels	origin vertices S	A	B	C	D
1 (S)	B	4				
	D	7				

B has potential 4 assigned from S.
D has potential 7 assigned from S.

Second iteration: label vertices reached directly only from S, B or D.

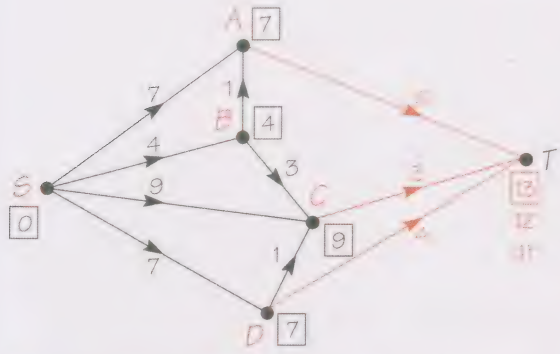


iteration	vertices assigned labels	origin vertices S	A	B	C	D
1 (S)	B	4				
	D	7				
2 (S,B,D)	A	7		5		
	C	9		7		8

A has potential 7 assigned from S.
C has potential 9 assigned from S.

2 SOLUTION CONTINUED

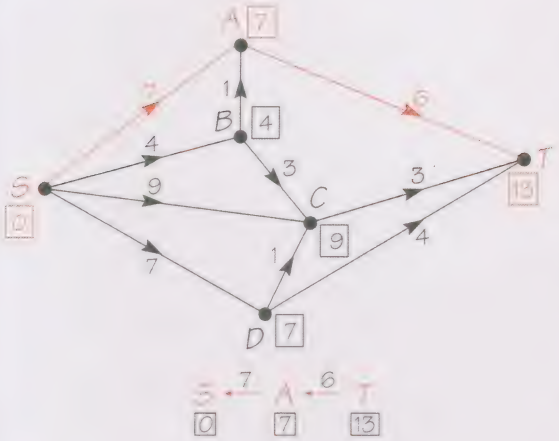
Third iteration: label vertices reached directly only from S,A,B,C or D.



iteration	vertices assigned labels	origin vertices S	A	B	C	D
1 (S)	B	4				
	D	7				
2 (S,B,D)	A	7		5		
	C	9		7		8
3 (S,A,B,C,D)	T		13		12	11

T has potential 13 assigned from A.

The longest path

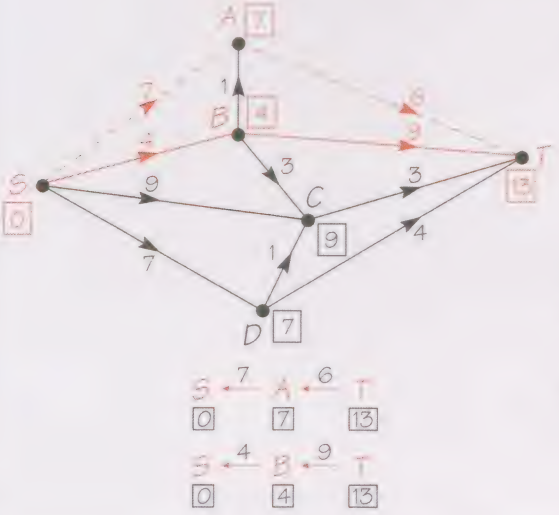


iteration	vertices assigned labels	origin vertices S	A	B	C	D
1 (S)	B	4				
	D	7				
2 (S,B,D)	A	7		5		
	C	9		7		8
3 (S,A,B,C,D)	T		13		12	11

Longest path is SAT with length 13.

Postscript: two longest paths

Add arc BT with length 9:



iteration	vertices assigned labels	origin vertices S	A	B	C	D
1 (S)	B	4				
	D	7				
2 (S,B,D)	A	7		5		
	C	9		7		8
3 (S,A,B,C,D)	T		13	13	12	11

Two longest paths: SAT and SBT with length 13.

TRACK 8

3 THE LONGEST PATH ALGORITHM: THE MAIN STEPS

START — Assign potential 0 to S.

GENERAL STEP — Consider all vertices which can be reached directly only from vertices with known potentials.

For each such vertex W, consider each arc VW into W, and assign W the label

(potential of V) + (distance VW)

unless W already has a *larger* label.

When all such arcs VW have been considered, make the vertex label at W a potential.

REPEAT — the GENERAL STEP with the new potential(s).

STOP — when T has been assigned a potential.

The longest distance from S to T *is* the potential of T.
To find a longest path, work backwards from T and include an arc VW wherever

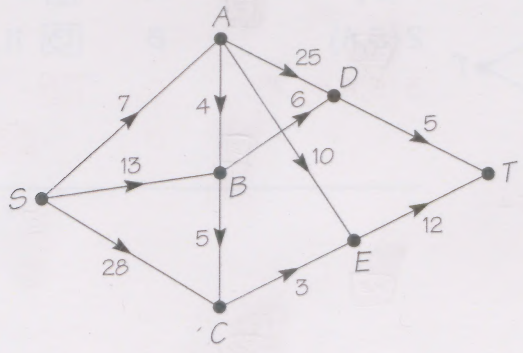
(potential of W) – (potential of V) = distance VW.

If the process does not terminate — for example, if the network has a cycle — then there is no longest path.

REMEMBER! — **shortest path algorithm**: *look* out from *vertices of known potential;*
longest path algorithm: *look* back to *vertices of known potential.*

4 PROBLEM

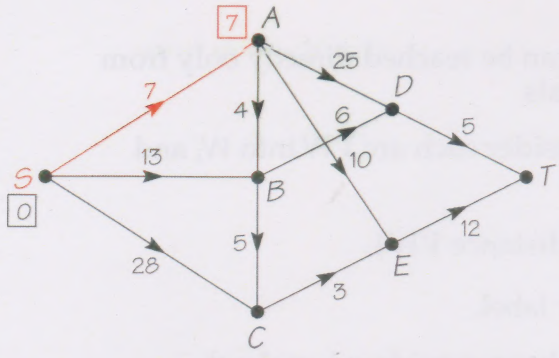
Find the longest path from S to T in the following network.



5 SOLUTION TO PROBLEM

Start by giving S zero potential.

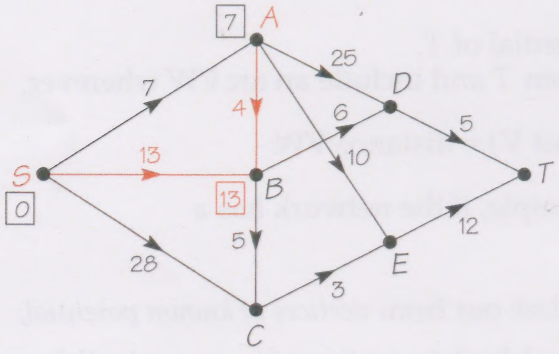
First iteration: label vertices reached directly only from S.



iteration	vertices assigned labels	origin vertices S	A	B	C	D	E
1 (S)	A	7					

A has potential 7 assigned from S.

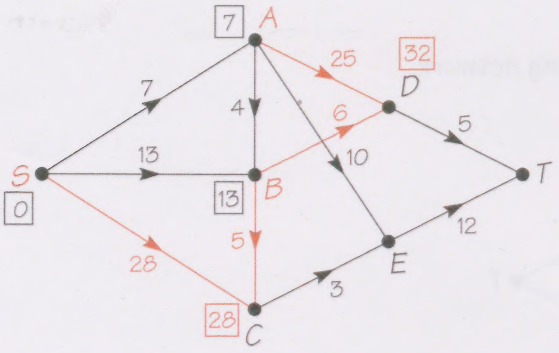
Second iteration: label vertices reached directly only from S or A.



iteration	vertices assigned labels	origin vertices S	A	B	C	D	E
1 (S)	A	7					
2 (S,A)	B	13	11				

B has potential 13 assigned from A.

Third iteration: label vertices reached directly only from S, A or B.

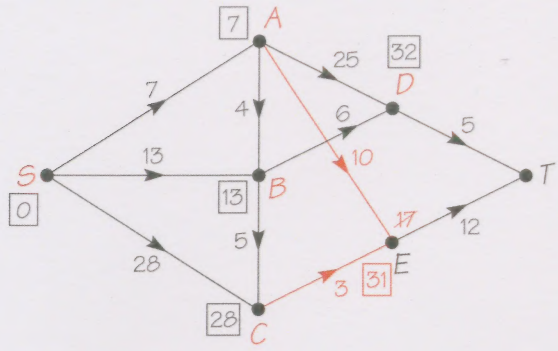


iteration	vertices assigned labels	origin vertices S	A	B	C	D	E
1 (S)	A	7					
2 (S,A)	B	13	11				
3 (S,A,B)	C	28		18			
	D		32	19			

C has potential 28 assigned from S.
D has potential 32 assigned from A.

5 SOLUTION CONTINUED

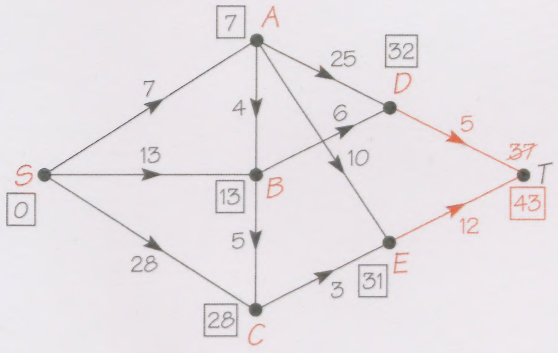
Fourth iteration: label vertices reached directly only from S, A, B, C or D.



iteration	vertices assigned labels	origin vertices S	A	B	C	D	E
1 (S)	A	7					
2 (S,A)	B	13	11				
3 (S,A,B)	C	28		18			
	D		32	19			
4 (S,A,B,C,D)	E		17		31		

E has potential 31 assigned from C.

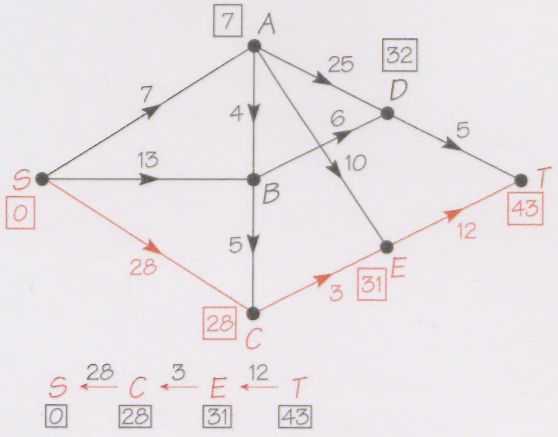
Fifth iteration: label vertices reached directly only from S, A, B, C, D or E.



iteration	vertices assigned labels	origin vertices S	A	B	C	D	E
1 (S)	A	7					
2 (S,A)	B	13	11				
3 (S,A,B)	C	28		18			
	D		32	19			
4 (S,A,B,C,D)	E		17		31		
5 (S,A,B,C,D,E)	T					37	43

T has potential 43 assigned from E.

The longest path



iteration	vertices assigned labels	origin vertices S	A	B	C	D	E
1 (S)	A	7					
2 (S,A)	B	13	11				
3 (S,A,B)	C	28		18			
	D		32	19			
4 (S,A,B,C,D)	E		17		31		
5 (S,A,B,C,D,E)	T					37	43

Longest path is SCET with length 43.

Printed in the United Kingdom.